NATGRID Documentation

****

************************** Overview of the CDAT interface to natgrid ************************

****

INTRODUCTION TO NGMATH

The ngmath library is a collection of interpolators and approximators for one−dimensional, two−dimensional and three−dimensional data. The packages, which were obtained from NCAR, are:

natgrid −− a two−dimensional random data interpolation package based on Dave Watson's nngridr.

dsgrid −− a three−dimensional random data interpolator based on a simple inverse distance weighting algorithm.

fitgrid −− an interpolation package for one−dimensional and two−dimensional gridded data based on Alan Cline's Fitpack. Fitpack uses splines under tension to interpolate in one and two dimensions.

csagrid −− an approximation package for one−dimensional, two−dimensional and three−dimensional random data based on David Fulker's Splpack. csagrid uses cubic splines to calculate its approximation function.

cssgrid −− an interpolation package for random data on the surface of a sphere based on the work of Robert Renka. cssgrid uses cubic splines to calculate its interpolation function.

shgrid −− an interpolation package for random data in 3−space based on the work of Robert Renka. shgrid uses a modified Shepard's algorithm to calculate its interpolation function.

COMPARISION OF NGMATH PACKAGES

Three−dimensional packages −− shgrid, csagrid and dsgrid.

shgrid is probably the package of choice for interpolation. It uses a least squares fit of biquadratics to construct its interpolation function. The interpolation function will pass through the original data points.

csagrid uses a least squares fit of cubic splines to calculate its approximation function: the calculated surface will not necesarily pass through the original data points. The algorithm can become unstable in data sparse regions.

dsgrid uses a weighted average algorithm and is stable in all cases, but the resultant interpolation is not usually smooth and execution time is very slow. dsgrid is probably best used when csagrid and shgrid fail or for comparative purposes.

Two−dimensional packages −− natgrid, fitgrid, csagrid and dsgrid.

natgrid is the package of choice in most cases. It implements a very stable algorithm and has parameters for adjusting the smoothness of the output surface.

fitgrid offers user−settable parameters for specifiying derivatives along the boundary of the output grid which are not available in natgrid.

csagrid produces an approximate two−dimensional surface which may be smoother than that produced by fitgrid
and natgrid.

dsgrid is not recommended for two−dimensional surfaces. natgrid is superior in all respects.

One−dimensional packages −− fitgrid and csagrid.

fitgrid is definitely the package of choice. It has many features not available in csagrid, such as interpolating parametric curves, finding integrals, handling periodic functions, allowing smoothing that varies from linear to a full cubic spline interpolation and specifying slopes at the end points.

Interpolation on a sphere −− cssgrid.

cssgrid is designed specifically for interpolating on a sphere. It uses cubic splines to calculate an interpolation function.

NATGRID PACKAGE

natgrid implements a natural neighbor interpolation method. The input for the interpolation is a set of randomly spaced two−dimensional coordinates with functional values at those coordinates; the output is a set of interpolated values at coordinates in a user specified rectangular grid. The coordinates in the output grid must be monotonic in each coordinate direction, but need not be evenly spaced. It is also possible to interpolate at a single point.

natgrid uses a weighted average method that is much more sophisticated than the inverse distance weighted average used by dsgrid. One distinguishing quality of natural neighbor interpolation is the way in which a set of neighboring points (the natural neighbor) is selected to use for interpolating at a point. The natural neighbor selection process avoids the problems common to methods based on choosing a fixed number
of neighboring points, or all points within a fixed distance. Another distinguishing quality of natural neighbor interpolation is the way that the weights are calculated for the functional values at the natural neighbor coordinates. These weights are based on proportionate area, rather than distances.

The method of finding the natural neighbors and calculating area−based weights to produce interpolated values is called natural neighbor linear interpolation. This produces an interpolation surface that has a continous slope at all points, except at the original input points. The result of natural neighbor linear interpolation can be visualized as producing a snugly fit sheet stretched over all of the input points.

The interpolation method in natgrid also allows for natural neighbor linear interpolation augmented by blending in gradient estimates. This is called natural neighbor nonlinear interpolation. It produces an interpolation surface that has a continuous slope at all locations; two tautness parameters can be set by the user to control the apparent smoothness of the output surface.

NATGRID CONTENTS

Access through Python to the natgrid package from NCAR's ngmath distribution is provided directly through the module
natgridmodule.so which was generated as a Python C language extension in order to export the natgrid functions
from the original C language library to Python.

REQUIRED FILE

natgridmodule.so –– the Python interface to the ngmath natgrid package.

USEFUL FILES

nat.py –– the object oriented interface including a general help package.
natgridtest.py –– the code to test nat.py and to write documentation.

USAGE

This module is designed to use in two ways. One is through the use of the object oriented interface to the underlying
functions. This approach is recommended for users not already familiar with the original natgrid distribtution because
it simplifies the calls to the routines. The other method uses the original functions calling them directly from Python.

––––––––––––––––––– OBJECT ORIENTED APPROACH ––––––––––––––––

The nat module contains the Natgrid class and its single method, rgrd, which provides access to all the natgrid functions. The object oriented approach has been organized as a two step process.

STEP 1.

To make an instance, r, type:

import nat

r = nat.Natgrid(xi, yi, xo, yo)
or
r = nat.Natgrid(xi, yi, xo, yo, listOutput = 'yes')

where xi, yi and xo, yo are the input and output grid coordinate arrays. The optional listOutput must
set to anything except 'no' if xo, yo are in list format as explained below. It is the responsibility
of the user to set listOutput if the output is in the list form.

The input grid must be organized in a list format always. The size of the xi array and the yi array are
necessarily equal. For example, if there are n randomly spaced input data points, there
are n values in xi and n values in yi.

There are two possible formats for the output grid. The output grid coordinate arrays may be a list like
the input array or it may be a rectangular grid. The choice between the two posibilities is made according
to requirements in subseqent calls to the method function. The first choice is required if the subsequent
call is to the single point mode interpolation. The list can have one or more points. Of course, the list
could describe a rectangular grid. For example, a rectangular grid with 10 x values and 20 y values can be

rewrtten in list form with 200 x value and 200 y values. However, this form requires calling the slower single point interpolator. The second choice is most efficient for the basic interpolation to a rectangular output grid. The output grid must be monotonic but need not be equally spced.

The grid coordinate arrays can be single precision (Numeric.Float32) or double precision (Numeric.Float64). The
decision on whether to call for a single or a double precision computation subsequently is made by looking at the type of these arrays.

To look at the default settings for the control parameters and a brief description of thier properties, type

r.printDefaultParameterTable()

To change a setting type the new value. For example, to set igr to 1, type

r.igr = 1

To find a value without printing the table, type the name. For example, to exam the value of hor, type

r.hor

To check the settings type

r.printInstanceParameterTable() –– prints in tabular form the parameters used in subsequent calls to the method
function rgrd.
or

printStoredParameters() –– prints the parameters in memory which may differ from the above if the user has made more than one instance of the Natgrid class.

STEP 2.

natgrid is restricted to two dimensions . Consequently, it is the user's responsibility to reduce the processing of higher dimensional data to a sequence of calls using only two dimensional data.

The computations are divided into two groups depending on whether the output arrays are in list form or in rectilinear
grid form. If they are in list format the single point mode is called to interpolate to those individual points. This is
the only process possible. On the other hand, if the output goes to a rectangular grid there are more choices. In addition to carrying out linear and nonlinear interpolations, it is possible to request aspects and slopes. The aspect
at a point on the interpolated surface is the direction of steepest descend. The slope is the value of the partial derivative taken in the direction of the aspect. The slope is measured as an angle that is zero in a horizonal surface
and positive below the horizontal.

The following examples cover the basic computations. They start with a indication of the appropriate STEP 1.

Example 1: the basic natural neighbor linear interpolation

As STEP 1 make an instance, r, with:

```
import nat

r = nat.Natgrid(xi, yi, xo, yo)
```

where the xo, yo grid is rectilinear as explained above in STEP 1.

Then call the primary interpolation computation to regrid the input data, dataIn, on the grid (xi, yi) to the output data, dataOut, on the grid (xo, yo), with

```
dataOut = r.rgrd( dataIn )
```

The computation is either single or double precision as determined by the precision submitted in the grid description in STEP 1.

It is also possible to request a wrap in the input grid and the input data in the longitude direction, assumed to be the yi grid coordinate, by adding a keyword as

```
dataOut = r.rgrd( dataIn, wrap = 'yes' )
```

Example 2: natural neighbor linear interpolation returning the aspect and the slope.

As STEP 1 make an instance, r, with:

```
import nat

r = nat.Natgrid(xi, yi, xo, yo)
```

where the xo, yo grid is rectilinear as explained above in STEP 1.

Then call the primary interpolation computation to regrid the input data, dataIn, on the grid (xi, yi) to the output data, dataOut, on the grid (xo, yo), while asking for the aspect and the slope on this output grid, with

```
dataOut, a, s = r.rgrd( dataIn, aspectSlope = 'yes' )
```

where a is the aspect, the direction of the steepest descent in degrees measured from 'north' and s is the slope in degrees measured from the horizontal. Necessarily, these are arrays aligned with the rectilinear output grid, xo, yo.

The computation is either single or double precision as determined by the precision submitted in the grid description in STEP 1.

It is also possible to request a wrap in the input grid and the input data in the longitude direction, assumed to be the yi grid coordinate, by adding a keyword as

```
dataOut, a, s = r.rgrd( dataIn, aspectSlope = 'yes', wrap = 'yes' )
```

Example 3: the basic natural neighbor nonlinear interpolation

The procedure for the nonlinear interpolation differs from the linear case in the need to set the control parameter igr. Follow Example 1 and insert the following statament after making the instance, r.

```
r.igr = 1
```

Example 4: natural neighbor nonlinear interpolation returning the aspect and the slope.

The procedure for the nonlinear interpolation differs from the linear case in the need to set the control parameter igr. Follow Example 2 and insert the following statament after making the instance, r.

```
r.igr = 1
```

Example 5: single point mode natural neighbor linear interpolation

As STEP 1 make an instance, r, with:

```
import nat

r = nat.Natgrid(xi, yi, xo, yo, listOutput = 'yes')
```

where the xo, yo output grid is in the list form (not a rectangular output grid) as explained above in STEP 1.

To call the single point mode interpolation computation to regrid the input data, dataIn, on the grid (xi, yi) to the output data, dataOut, on the grid (xo, yo), type

```
dataOut = r.rgrd( dataIn )
```

The computation is either single or double precision as determined by the precision submitted in the grid description in STEP 1. In the single point mode it is not possible to request the aspect and the slope.

Example 6: single point mode natural neighbor nonlinear interpolation

The procedure for the nonlinear interpolation differs from the linear case in the need to set the control parameter igr. Follow Example 5 and insert the following statament after making the instance, r.

```
r.igr = 1
```

−−−−−−−−−−−−−−−−−−− ORIGINAL FUNCTION APPROACH −−−−−−−−−−−−−−−−−

The module natgridmodule.so exports the following functions to Python from the original ngmath C library:

Single precision procedures:

natgrids − primary function for gridding.
seti − set int parameter values.
geti − retrieve values for int parameters.
setr − set float parameter values.
getr − retrieve values for float parameters
setc − set char parameter values.

getc – retrieve values for char parameters.
getaspects – get aspect values, if calculated by setting sdi = 1.
getslopes – get slope values, if calculated by setting sdi = 1.
pntinits – initiate single point mode.
pnts – interpolate at a single point.
pntend _ terminate single point mode.

Double precision procedures:

natgridd – primary function for gridding.
setrd – set float parameter values.
getrd – retrieve values for float parameters
getaspectd – get aspect values, if calculated by setting sdi = 1.
getsloped – get slope values, if calculated by setting sdi = 1.
pntinitd – initiate single point mode.
pntd – interpolate at a single point.
pntendd _ terminate single point mode.

Information on the use of the routines is available by importing natgridmodule and printing the docstring of interest. For example, documentation for the routine natgrids is obtained by typing

```
import natgridmodule
print natrridmodule.natgrids.__doc__
```

This same information is available in the help package.

A description of the control parameters is not in the natgridmodule documentation. It can be found by typing

```
import nat
nat.printParameterTable()
```

The documentation associated with the natgridmodule.so, such as the doctrings, describe the C code.

DOCUMENTATION

Documentation is provided through Python's docstrings, essentially Python style program comments. A help package provides instructions on the use of the natgrid module. A table of contents is printed to the screen by typing

```
nat.help()
```

after importing nat.

A hard copy of all the pertinent 'docstring' documentation written to the file natgridmodule.doc can be produced by typing

```
nat.document()
```

As an alternate to using the help package, online documentation for the natgrids function, for example, is available directly from the natgrids doctring by typing

import natgridmodule

print natgridmodule.natgrids.__doc__

TESTING

To run a test of the natgrid computations and to get a copy of this documentation, type

cdat natgridtest.py

HELP PACKAGE EXAMPLE

************************ Default Parameter Table **********************

----
name type legal value defau description
---- ----- ------------ -------
----
adf int 0 = no or 1 0 produce data file of algoritmic info for display? (see alg)
alg char any file nam "nnalg file name for algoritmic display tool (see adf)
asc int 0 = no or 1 1 is automatic scaling is allowed?
bI float >= 1. 1.5 tautness increasing effect of the gradients by increasing bI
bJ float >= 1. 7.0 tautness decreasing breadth of region affected by gradients
dup int 0 = yes or 1 1 are duplicate input coordinates are allowed?
ext int 0 = no or 1 1 is extrapolation allowed outside the convex hull?
hor float >= 0. −1.0 amount of horizontal overlap from outside current region
igr int 0 = no or 1 0 are gradients are to be computed?
magx float > 0. 1.0 scale factor for x coordinate values
magy float > 0. 1.0 scale factor for y coordinate values
magz float > 0. 1.0 scale factor for z coordinate values
non int 0 = yes or 1 0 are interpolated values are allowed to be negative?
nul float any float 0.0 value for points outside the convex hull if no extrapolation
rad int 0 = rad or 1 0 are slopes and aspects are returned in radians or degrees?
sdi int 0 = no or 1 0 are slopes and aspects to be computed?
upd int 0=N to S or 1 does output array from giving N to S or S to N?
ver float >= 0. −1.0 amount of vertical overlap from outside current region
xas float > 0. 0.0 scale used by automatic scaling of x in last interpolation
yas float > 0. 0.0 scale used by automatic scaling of y in last interpolation
zas float > 0. 0.0 scale used by automatic scaling of z in last interpolation.